AFCRL 64-411

AUTOMATED LOGIC FOR SEMI-AUTOMATED MATHEMATICS

James R. Guard

APPLIED LOGIC CORPORATION
20 Nassau Street
Princeton, New Jersey

Contract No. AF19 (628)-3250

Project No. 5632

Task No. 563205

Scientific Report No. 1

March 30, 1964

Prepared
for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

AFCRL 64-411

AUTOMATED LOGIC FOR SEMI-AUTOMATED MATHEMATICS

James R. Guard

APPLIED LOGIC CORPORATION
20 Nassau Street
Princeton, New Jersey

Contract No. AF19 (628)-3250

Project No. 5632

Task No. 563205

Scientific Report No. 1

March 30, 1964

Prepared
for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

Requests for additional copies by Agencies of the Department of Defense, their contractors, and other Government agencies should be directed to the:

DEFENSE DOCUMENTATION CENTER (DDC)
CAMERON STATION
ALEXANDRIA, VIRGINIA 22314

Department of Defense contractors must be established for DDC services or have their 'need-to-know' certified by the cognizant military agency of their project or contract.

All other persons and organizations should apply to the:

U. S. DEPARTMENT OF COMMERCE
OFFICE OF TECHNICAL SERVICES
WASHINGTON 25, D. C.

ABSTRACT

UNCLASSIFIED

In previous work with our colleagues, we have investigated some of the possibilities of proving mathematical theorems on a computer on a man-machine basis. At the intermediate stages in a proof we are, in general, trying to prove some formula from certain suppositions and previously proved theorems. If such intermediate steps have "trivial" proofs, we might hope to have the machine verify this automatically. This report describes some algorithms which verify certain "trivial" proofs. These algorithms can be read off from the definition of proof in the formal systems S_1 through S_6 described in this report. Algorithms concerning the propositional connectives are explicated by systems S_1 and S_2; quantifiers by S_3 and S_4; ω-order predicate-function calculus by S_5; and many sorted variables and constants for ω-order predicate-function calculus by S_6.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the contributions of his colleagues, James H. Bennett, William B. Easton, and Thomas H. Mott, to this report. As a team, we developed the SAM programs; and though the development of the automatic logic was the author's personal responsibility, he had many helpful discussions with his colleagues which bear on this report.

The author is affiliated with the Mathematics Department of Princeton University and wishes to acknowledge their encouragement in this study in bridging mathematics and computing.

Finally, thanks to the staff at Applied Logic for administrative assistance.

TABLE OF CONTENTS

SUMMARY

This report defines and describes six formal systems of logic, S_1 through S_6, for which proof procedures or partial proof procedures are readily contrived. The systems S_1 through S_4 are considered in order to simplify the descriptions of S_5 and S_6. System S_1 is a fragment of the classical propositional calculus whose theorems are those tautologies which can be shown tautologous by assuming them to take on the value falsehood and arriving at an inconsistent assignment to the variables by not using any "branching" rules. This system suggests an efficient means of handling the propositional connectives in the later systems. System S_2 is the completion of S_1. S_2 has "branching" rules which correspond to treating certain propositional variables by cases. This differs from Gentzen's treatment by Sequenzen in that Gentzen's "branching" rules consider the value of the antecedent or consequent of a formula by cases. System S_3 is a fragment of the first order predicate-function calculus. In S_3, formulas are proved by contradiction. The quantifiers of the denial are stripped by putting the denial in miniscope form and replacing them by Skolem functors. This is reminiscent of the Herbrand technique. However S_3 uses a process called matching to consider only reasonable Herbrand disjuncts. System S_4 is the completion of S_3 obtained by carrying out the full Herbrand process. The completeness theorem for S_4 is outlined in an appendix. System S_5 is an ω-order predicate-function calculus patterned after

S_4. A method for handling types is included in S_5 which allows us to prove theorems at the lowest possible type and yet have instances of these theorems involving higher types available by substitution. Function-like properties of predicates can also be obtained as instances of more general theorems concerning functions by substitution. System S_6 is an extension of S_5 to a many-sorted ω - order predicate-function calculus. It is believed that many of the problems attendent to mechanizing a many-sorted calculus are resolved by S_6. A careful treatment of the matching process has not however as yet been carried out for S_6. However no insurmountable obstacles are expected.

0. Introduction

It is the opinion of this writer and his colleagues (J. H. Bennett, W. B. Easton, and T. H. Mott, Jr.) at the Applied Logic Corporation, that no computers in the near future will be powerful enough to carry out any non-trivial deductive mathematics. This does not preclude the possibility of doing serious mathematics by real-time operator-mathematician intervention. Such a program we have dubbed semi-automated mathematics (SAM). The purpose of this report is to describe some algorithms which are useful for the automated portion of SAM. These algorithms can be "read-off" from the definition of proof in the formal systems S_1 through S_6 described in Sections 1-6.

Some history is in order. We quickly realized that the method of mathematical logic known as natural deduction by subordinate-proof (cf. Fitch [1] or Jaskowski [2]) is a convenient way to display mathematical proofs. This convenience holds for both the man and the machine. Two programs were written for the IBM 1620 using this method. The first program, SAM I (cf. [3]), handled the propositional calculus on a semi-automated (man-machine) basis. The second program, SAM III (cf. [4]), treated semi-automatic ω-order predicate-function calculus with equality and methods of definition.

A method called matching for efficiently applying (upon command of the man) rules of inference, definitions, axioms, and previously proved theorems was developed for another IBM 1620 program, SAM II.

SAM II (cf. [5]) was a system for semi-automatic first-order, quantifier-free, (constant) function, axiomatics with equality and methods of definition. Matching was also used in SAM III. In SAM III we added some automatic techniques of the following form. In the middle of an incomplete subordinate proof we may be attempting to prove a formula B from previously proved theorems and relevant hypotheses $C_1, C_2, \ldots, C_n$. The machine attempts to prove this automatically by creating the set S consisting of $C_1, \ldots, C_n$ and some denial of B, and then trying to reduce S to a contradiction. Matching is used in the reduction of S. Also certain equality rules were automated for this reduction. The method of handling the propositional connectives in this reduction is explicated by the systems S_1 and S_2 below. The method of stripping quantifiers, called Skolemization, for the reduction of S, along with matching are described in S_3. S_4 is the completion of S_3 to the full first order predicate-function calculus without equality. S_5 extends S_4 to the ω-order predicate-function calculus and includes the rudiments of type theory. The main difficulty in extending to S_5 is in giving an adequate definition of matching. If portions of matching in S_5 were restricted and some equality rules were added to S_5, we would have approximately the automatic portion of SAM III.

The system S_6 is tendered as a solution of the "sorted-symbol" problem. In intuitive mathematics we use certain personal conventions concerning variables ranging over certain mathematical objects. For

example, m, n, i, j frequently range over integers; p, q over primes; r, s, t over rationals; x, y over reals; G over groups; H over sub-groups; etc. Also we connect various related objects in a mathematic structure by the process of identication. E. g. if G is a group with composition f, then we might use H_i to range over subgroups of G. However with no qualms at all we call the composition of each of these subgroups, f. After all, only a purist considers the restriction of a function to be a different function. Unfortunately, computers are purists. Also we want theorems concerning rational numbers to be available to integers without intervention by the man. All these problems, at least theoretically, are resolved by S_6.

No attempt has been made to consider automatic equality rules in this report. Hopefully, this can be the subject of a later report. Handling the equality rules in a manner which allows us to do "trivial" mathematics automatically may be the last serious theoretical hurdle before some serious mathematics can be developed on a computer using SAM.

Notation. Throughout this report we shall use the following notations and conventions.

1) $\vdash_i$ will denote provability in system S_i.

2) B, C, D, B_1, C_1, D_1, B_2, ... will be metamathematical symbols ranging over the well-formed formulas (w ffs)

or well-formed terms (wfts) of a given system.

3) Formulas such as $B(C_1, \ldots, C_n)$ will denote a wff where attention is drawn to all occurrences of the wffs $C_1, \ldots, C_n$ which appear as subformulas of B.

4) Formulas such as $B(\ldots C \ldots)$ will denote a wff where attention is drawn to a particular single occurrence of a subformula C of B.

5) The usual rules for associating the connectives, dropping parentheses, or replacing pairs of parentheses by dots will be observed.

6) Frequently we will not explicitly name certain variables but only give names to metamathematic symbols which range over the variables of a given system.

7) After the final clause of an inductive definition in the metalanguage, consider as added a clause reading that an object is in the class defined if its being so follows from a finite number of applications of the above clauses. E.g. in the definition of wft for S_1 we understand as present a fourth clause reading.

"4) A string of primitive symbols of S_1 is a well-formed formula of S_1, only if its being so follows from finitely many applications of 1), 2), and 3) above."

1. The Formal System S_1

The system S_1 formalizes a fragment of the propositional calculus. By itself S_1 is of little interest. However S_1 lends itself to efficient mechanization and is a sub-system of the more powerful systems S_2, S_3, S_4, S_5, and S_6.

Primitive symbols of S_1:

1) Propositional constants T F

2) Propositional connectives & v $\supset$ $\equiv$ $\sim$

3) Improper symbols) (

4) An infinite list of propositional variables. We shall use p, q, r, p_1, ... as metamathematical names ranging over the propositional variables.

5) We use B, C, B_1, C_1, B_2, ... to range over wffs of S_1.

Well-formed formulas (wffs) of S_1:

1) A propositional variable standing alone is a wff.

2) T and F are wffs.

3) If B and C are wffs then so are $\sim$B, (B v C), (B & C), (B $\supset$ C), and (B $\equiv$ C).

Theorems of S_1 . A finite sequence $\{B_1^{(1)}, \ldots, B_{n_1}^{(1)}\}, \ldots,$ $\{B_1^{(r)}, \ldots, B_{n_r}^{(r)}\}$ of finite sets of wffs of S_1 is said to be a proof of the theorem B if

1) $n_1 = 1$ and $B_1^{(1)}$ is $\sim B$,

2) some $B_i^{(r)}$ is F , and

3) some $B_i^{(j)}$ is p (resp. $\sim p$) and $\{B_1^{(j+1)}, \ldots, B_{n_{j+1}}^{(j+1)}\}$ is obtained from the previous set by deleting the $B_k^{(j)}$'s which are p (resp. $\sim p$) and replacing all the occurrences of p in the remaining set by T (resp. F) , or $\{B_1^{(j+1)}, \ldots, B_{n_{j+1}}^{(j+1)}\}$ is obtained from the previous set by replacing one (or two) of the $B_i^{(j)}$'s by wffs using one of the reduction rules below.

Reduction rules for S_1 :

1) Replace B & C by B and C.

2) Replace $\sim(B \supset C)$ by B and $\sim C$.

3) Replace $\sim(B \vee C)$ by $\sim B$ and $\sim C$.

4) Replace any of $B(\ldots\sim\sim C\ldots)$, $B(\ldots T \supset C \ldots)$, $B(\ldots F \vee C \ldots)$, $B(\ldots C \vee F \ldots)$, $B(\ldots T \,\&\, C \ldots)$, $B(\ldots C \,\&\, T \ldots)$, $B(\ldots T \equiv C \ldots)$, or $B(\ldots C \equiv T \ldots)$ by $B(\ldots C \ldots)$.

5) Replace any of $B(\ldots C \supset F \ldots)$, $B(\ldots F \equiv C \ldots)$, or $B(\ldots C \equiv F \ldots)$ by $B(\ldots \sim C \ldots)$.

6) Replace any of $B(\ldots F \supset C \ldots)$, $B(\ldots C \supset T \ldots)$, $B(\ldots T \vee C \ldots)$, $B(\ldots C \vee T \ldots)$, or $B(\ldots \sim F \ldots)$ by $B(\ldots T \ldots)$.

7) Replace any of $B(\ldots F \,\&\, C \ldots)$, $B(\ldots C \,\&\, F \ldots)$, or $B(\ldots \sim T \ldots)$ by $B(\ldots F \ldots)$.

By way of example we prove some theorems in S_1 :

$\vdash_1 T$

Proof: $\{\sim T\}$, $\{F\}$.

$\vdash_1 p \supset p$

Proof: $\{\sim .\ p \supset p\}$, $\{p, \sim p\}$, $\{\sim T\}$, $\{F\}$. Alternatively we could use the proof $\{\sim .\ p \supset p\}$, $\{p, \sim p\}$, $\{F\}$.

$\vdash_1 p \supset .\ q \supset p$

Proof: $\{\sim .\ p \supset .\ q \supset p\}$, $\{p, \sim .\ q \supset p\}$, $\{p, q, \sim p\}$, $\{F, q\}$.

$\vdash_1 p \supset (q \supset r) \supset .\ p \supset q \supset .\ p \supset r$

Proof: $\{\sim .\ p \supset (q \supset r) \supset .\ p \supset q \supset .\ p \supset r\}$, ..., $\{p \supset (q \supset r), p \supset q, p, \sim r\}$, $\{T \supset (q \supset r), T \supset q, \sim r\}$, $\{q \supset r, T \supset q, \sim r\}$, $\{q \supset r, q, \sim r\}$, $\{T \supset r, \sim r\}$, $\{r, \sim r\}$, $\{F\}$.

$\vdash_1 \quad \sim p \supset \sim q \supset . \; q \supset p$

Proof: $\{\sim . \sim p \supset \sim q \supset . \; q \supset p\}, \ldots .$

$\{\sim p \supset \sim q , \quad q , \quad \sim p\}, \quad \{\sim p \supset \sim T , \sim p\},$

$\{\sim p \supset F , \quad \sim p\} , \{\sim \sim p , \sim p\}, \ldots ., \{F\} .$

Remark: As is well known, the last three theorems are frequently taken as the axioms along with modus ponens and substitution for a development of the propositional calculus in terms of only $\supset$ and $\sim$. However we do not have modus ponens as a rule or as a derived rule of S_1 since for example $p \supset p \supset q \supset . \; p \supset p \supset r \supset . q \supset \sim r$ is tautologous but not a theorem of S_1. In fact substitution is not a derived rule of S_1 since $\vdash_1 p \supset p$, while $p \equiv q \supset . \; p \equiv q$ is not a theorem. Notice that no wffs of the forms $B \,\&\, C$, $B \equiv C$, $\sim (B \supset C)$, and $\sim (B \vee C)$ are theorems of S_1. While not every tautology is a theorem we have the converse:

Theorem. Every theorem of S_1 is a tautology.

Proof: Let $\{B_1, \ldots ., B_m\}$ be obtained from $\{C_1 , \ldots ., C_n\}$ by one application of the reduction rules. Then $(B_1 \,\&\, \ldots \,\&\, B_m) \equiv (C_1 \,\&\, \ldots \,\&\, C_n)$ is tautologous as can be seen by examining each of the reduction rules. Hence if B is a theorem of S_1 we have $\sim B \equiv F$, and therefore B, is tautologous.

Remark. Notice that the theorems of S_1 are exactly those tautologies B which can be proved tautologous by assigning F to B and arriving at a contradiction using only the uniquely determined lines of the truth-tables of the connectives and not treating any propositional variables or subformulas by cases.

2. The Formal System S_2

The system S_2 is obtained from S_1 by modifying the definition of proof of a theorem.

Theorems of S_2 : A finite sequence $B_1, \ldots, B_n$ of wffs of S_1 is said to be a proof of B_n in the system S_2 if for each i ($i = 1, 2, \ldots, n$) B_i is a theorem of S_1 or there are j and k ($1 \leq j < i$ and $1 \leq k < i$) such that B_j is $p \supset B_i$ and B_k is $\sim p \supset B_i$.

Examples:

$\vdash_2 \quad p \equiv p$

Proof: $\vdash_1 p \supset . p \equiv p$, by $\{\sim . p \supset . p \equiv p\}$, $\{p, \sim . p \equiv p\}$, $\{\sim . T \equiv T\}$, $\{\sim T\}$, $\{F\}$. Also $\vdash_1 \sim p \supset . p \equiv p$, by $\{\sim . \sim p \supset . p \equiv p\}$, $\{\sim p, \sim . p \equiv p\}$, $\{\sim . F \equiv F\}$, $\{\sim\sim F\}$, $\{F\}$. Hence $p \supset . p \equiv p$, $\sim p \supset . p \equiv p$, $p \equiv p$ is a proof of $p \equiv p$ in S_2.

$\vdash_2 \quad p \supset . p \supset q \equiv q$

Proof: It is easily checked that first four wffs of the following sequence are theorems of S_1 :

$p \supset . q \supset . p \supset . p \supset q \equiv q$, $\sim p \supset . q \supset . p \supset . p \supset q \equiv q$,

$p \supset . \sim q \supset . p \supset . p \supset q \equiv q$, $\sim p \supset . \sim q \supset . p \supset . p \supset q \equiv q$,

$q \supset .\ p \supset .\ p \supset q \equiv q$, $\sim q \supset .\ p \supset .\ p \supset q \equiv q$,

$p \supset .\ p \supset q \equiv q$. Hence this sequence is a proof of

$p \supset .\ p \supset q \equiv q$ in S_2 .

This proof is unnecessarily long. Suppose we attempt to prove this wff in S_1 . We have $\{\sim p \supset .\ p \supset q \equiv q\}$, $\{p, \sim p \supset q \equiv q\}$, $\{\sim .\ T \supset q \equiv q\}$, $\{\sim .\ q \equiv q\}$ but we can proceed no further. However, we observe that we need only treat this last set by cases on q. This suggests the following shorter proof of the original wff in S_2 : $q \supset .\ p \supset .\ p \supset q \equiv q$, $\sim q \supset .\ p \supset .\ p \supset q \equiv q$, $p \supset .\ p \supset q \equiv q$ where indeed the first two wffs are theorems of S_1 as can be easily checked.

The longer, "canonical" proof of the example immediately above suggests the following completeness theorem for S_2 :

Theorem. The theorems of S_2 coincide with the tautologous wffs of S_2 .

Proof: That every theorem of S_2 is tautologous is easily observed from the facts that the theorems of S_1 are tautologous and if $p \supset B$ and $\sim p \supset B$ are tautologous then so is B .

On the other hand suppose $B(p_1, \ldots, p_n)$ is tautologous and contains only the propositional variables $p_1, \ldots, p_n$. Then $\tilde{p}_1 \supset .\ \tilde{p}_2 \supset .\ \ldots \supset .\ \tilde{p}_n \supset B$ where $\tilde{p}_i$ is either p_i or $\sim p_i$, will be provable in S_1 with the proof $\{\sim .\ \tilde{p}_1 \supset .\ \ldots \supset .\ \tilde{p}_n \supset B\}$, $\ldots$, $\{\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_n, \sim B\}$, $\{\tilde{p}_2, \ldots, \tilde{p}_n, \sim B(\tau_1, p_2, \ldots, p_n)\}$, $\ldots$,

$\{\sim B(\tau_1, \tau_2, \ldots, \tau_n)\}, \ldots, \{\sim T\}, \{F\}$,

where τ_i is T or F according as $\hat{p}_i$ is p_i or $\sim p_i$. Now combining the 2^n different theorems of S_1 of the form $\tilde{p}_1 \supset . \tilde{p}_2 \supset \ldots\ldots \supset . \tilde{p}_n \supset B$ we obtain B as theorem of S_2 from a proof with $2^{n+1} - 1$ wffs in the obvious fashion.

The shorter proof of $p \supset . p \supset q \equiv q$ suggests that frequently we can find a proof of a tautology with n propositional variables, which has a length considerably shorter than the $2^{n+1} - 1$ steps needed in the "canonical" proof. This is a common occurrence in mathematics; but all too often some collection of heuristic shortcuts that a mathematician uses can not be formalized. However, in this case we have constructed S_1 and S_2 so that the shortcut is easily mechanized. This is described by the following definition, theorems, and examples:

Definition. Let B* be the last set of a finite sequence of finite sets of wffs satisfying 1), 3), and 4) of the definition of a theorem of S_1 and such that B* either contains F or, it contains no formulas of the forms p or $\sim p$ and no reduction rule is applicable to B*. For definiteness let us suppose further that in this sequence of sets, 3) was applied where possible before 4); and where more than one application of 3) (resp. 4)) was possible the application first effected was the one which eliminated the left-most p or $\sim p$ (respectively, connective), considering the sets as ordered sets in the natural way.

Theorem. Let B be a tautology. Let m be the number of distinct variables in B^*. Then there is a proof of B in S_2 of length $2^{m+1} - 1$.

Proof: Let $q_1, q_2, \ldots, q_m$ be the m variables in B^*. Then clearly $\tilde{q}_1 \supset . \ldots . \supset . \tilde{q}_m \supset B$ is provable in S_1 where $\tilde{q}_i$ is q_i or $\sim q_i$. Now proceed as in the preceding theorem.

Examples.

$\vdash_2 \quad p \supset . p \supset q \equiv q \qquad (B)$

Proof: Now B^* is $\{\sim . q \equiv q\}$ so that $q \supset . p \supset . p \supset q \equiv q$, $\sim q \supset . p \supset . p \supset q \equiv q$, $p \supset . p \supset q \equiv q$ is the proof of length $2^{1+1} - 1 = 3$ indicated by the last theorem. From a computational standpoint it is much more efficient to combine the proofs of $q \supset . p \supset . p \supset q \equiv q$ and $\sim q \supset . p \supset . p \supset q \equiv q$ in S_1 along with the proof of length 3 in S_2 into the following computational scheme:

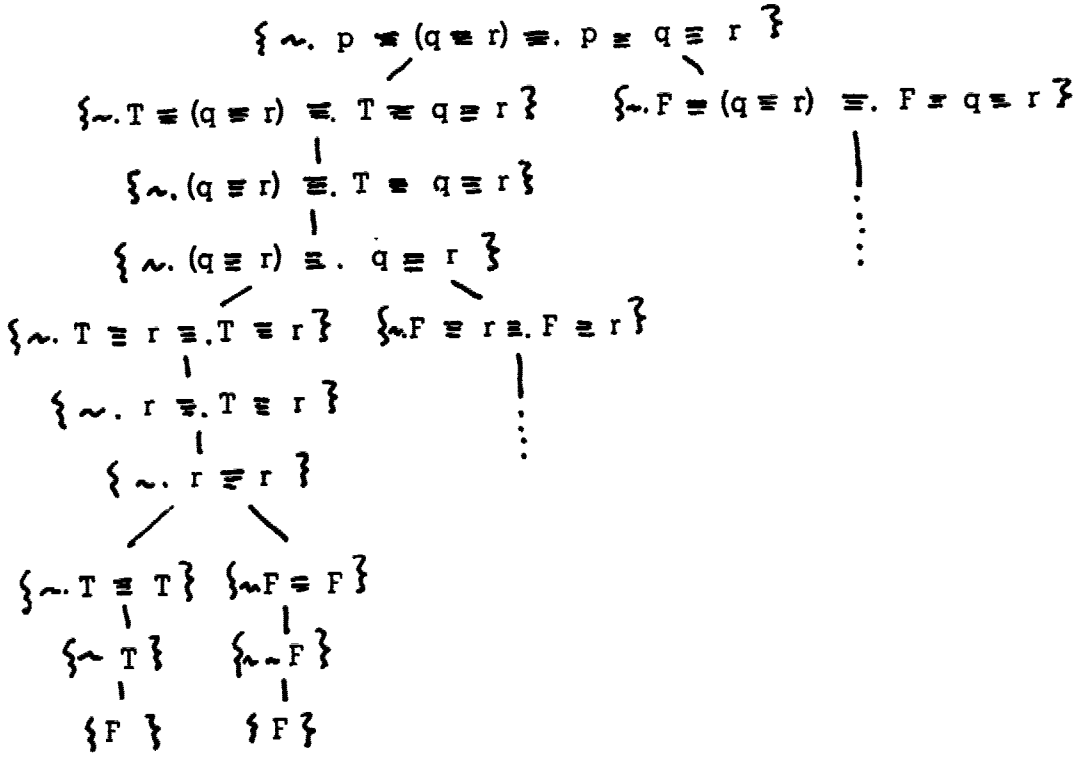
$$\{\sim . p \supset . p \supset q \equiv q\}$$
$$\{p, \sim p \supset q \equiv q\}$$
$$\{\sim . T \supset q \equiv q\}$$

$$\{\sim . T \supset T \equiv T\} \qquad \{\sim . T \supset F \equiv F\}$$
$$\{\sim . T \equiv T\} \qquad \{\sim . F \equiv F\}$$
$$\{\sim T\} \qquad \{\sim \sim F\}$$
$$\{F\} \qquad \{F\}$$

As is frequently the case with branching structures, this display can be efficiently mechanized by a push-down list arrangement.

$\vdash_2 \quad p \equiv (q \equiv r) \quad \equiv . \quad p \not\equiv q \equiv r$

Proof: This tautology generates the following computational scheme:



Note that in this case no rules from S_1 are applicable so that our computational scheme degenerates into the truth-table procedure. This is, however, preferable to proving eight longer theorems in S_1 and combining them to make a proof in S_2 with fifteen steps.

Comparison with Gentzen's algorithms for the propositional calculus.

One of the most efficient algorithms for the propositional calculus arises from Gentzen's system of natural deduction (cf. Kleene's [6] system G1, p. 442). The reduction rules 1), 2), and 3) correspond respectively to $\& \rightarrow$, $\rightarrow \supset$, and $\rightarrow \vee$. The special case of $B(\ldots C\ldots)$ replacing $B(\ldots \sim\sim C\ldots)$ where $B(\ldots C\ldots)$ is just C corresponds to $\rightarrow \sim$ followed by $\sim \rightarrow$. Notice that all of these rules are non-branching rules of G1. Also certain applications of the branching rules of G1 are essentially non-branching. E.g. $\supset \rightarrow$, where the antecedent of the implication to be eliminated is also a hypotheses could be replaced by a special non-branching rule (for $\{D, D \supset C\}$ is equivalent to $\{D, C\}$). This corresponds roughly to the reduction rule:

"replace $B(\ldots T \supset C \ldots)$ by $B(\ldots C\ldots)$."

On one hand our reduction rule is more general since C may be buried quite deeply in $B(\ldots C\ldots)$, while on the other hand we may have the formulas D and $D \supset C$ and not be able to get $T \supset C$ from this in S_1 (e.g. $p \supset p$ and $p \supset p \supset C$). We do not choose to modify S_1 to include some rule which would get us from $\{D, D \supset C\}$ to $\{D, T \supset C\}$ for reasons of computational efficiency.

In essence then S_1 does almost all possible non-branching inferences thus eliminating much of the branching necessary in G1. Only as a last resort -- thus invoking S_2 -- are branching-type rules

used. And in fact the branching used in S_2 tends to eliminate further branching. Of course for some examples our method corresponds almost exactly to the truth-table method. E.g. see the proof of $p \equiv (q \equiv r) \equiv . \; p \equiv q \equiv r$ in the example above. However the natural deduction method is similarly a disaster in these freak cases. In general our method seems to be more efficient than natural deduction which in general is considerably more efficient than the direct truth-table method. And further, the method of S_2 lends itself very nicely to extension to the higher order calculi.

Considering again the remark at the end of Section 1, the method of S_2 corresponds exactly to the method of assigning F to a tautology and using the uniquely determined lines of the truth-tables and then by cases on the propositional variables when necessary, in order to derive a contradiction. On the other hand, the cut-free version of G1 corresponds exactly to the method of assigning F to a tautology and using the uniquely determined lines of the truth-tables and then by cases on the two halves of a subformula using the other lines of the truth-table. (E.g., $A \supset B$ is T is contradictory, if and only if, A is F is contradictory and B is T is contradictory.)

3. The Formal System S_3.

The system S_3 is a fragment of the first order predicate calculus with symbols for both predicates and functions but without equality. This system is introduced so that the techniques for handling quantifiers and the instantiating of hypotheses can be explained in the simplest possible setting. The propositional portion of S_3 is an evident generalization of S_1. Techniques for handling the equality rules will be discussed later.

Primitive symbols of S_3 :

1) - 3) As in S_1, with the comma as an additional improper symbol.

4) Quantifiers A E

5) An infinite list of individual variables.

6) An infinite list of individual constants.

7) An infinite list of predicate constants of each degree n (n = 1 , 2 , 3 , ...) .

8) An infinite list of function constants of each degree n (n = 1 , 2 , 3 , ...) .

Remark: We use the following symbols in the metalanguage of S_3 :

1) individual variables $x \quad y \quad z \quad x_1 \quad y_1 \quad \ldots$

2) individual constants $a \quad b \quad c \quad d \quad e \quad a_1 \quad \ldots$

3) n-ary predicate constants $P^{(n)} \quad Q^{(n)} \quad R^{(n)} \quad P_1^{(n)} \quad Q_1^{(n)} \quad \ldots$

4) n-ary function constants $G^{(n)}$ $H^{(n)}$ $I^{(n)}$ $G_1^{(n)}$ $H_1^{(n)}$ ....

5) well-formed terms M N M_1 N_1 M_2 ....

6) well-formed formulas B C D B_1 C_1 ....

When no ambiguity arises we will drop the superscripts on the predicates and functions.

Well-formed expressions of S_3 :

1) Individual variables and individual constants are well-formed terms (wfts).

2) If $M_1, \ldots, M_n$ are wfts and $G^{(n)}$ is an n-ary function constant then $G^{(n)}(M_1, \ldots, M_n)$ is a wft.

3) If $M_1, \ldots, M_n$ are wfts then $P^{(n)}(M_1, \ldots, M_n)$ is a (atomic) well-formed formula (wff).

T and F are (atomic) wffs.

The variables occurring in atomic wffs are said to occur free.

If B and C are wffs then so are $(B \supset C)$, $(B \ \& \ C)$, $(B \vee C)$, $(B \equiv C)$, and $\sim B$. The variables which free in B and C are free in $(B \supset C)$, $(B \ \& \ C)$, $(B \vee C)$, and $(B \equiv C)$. The variables free in B are free in $\sim B$. If B is a wff, so is $(Ax)B$ and $(Ex)B$. Occurrences of x in $(Ax)B$ are called bound occurrences of x and those occurrences of x which are free in B are said to be bound by the initial quantifier of $(Ax)B$ or $(Ex)B$. Free variables of B other than x are free variables of

(Ax) B and (Ex) B which are said to be <u>in the scope</u> of the initial quantifier.

In S_1 we proved a wff B by attempting to derive F from $\sim B$. In S_3 we proceed analogously--however, in place of the negation of a wff we use a quantifier-free wff which contains the intent of the denial of the wff. To this end we define a quantifier-free form of a wff.

<u>Definition</u>. Let B^0 -- called the <u>Skolemization</u> of B (where B is a wff containing no free variables) -- be the formula obtained from B, by successively applying the following reduction rules (the applicable rule appearing earliest in the list is applied first):

1) Replace subformulas of the form $\sim\sim C$, $\sim(C \supset D)$, $\sim(C \vee D)$, $\sim(C \,\&\, D)$, $C \supset D$, $C \equiv D$, $\sim(C \equiv D)$, $\sim(Ax)\,B$, and $\sim(Ex)\,B$ respectively by C, $C \,\&\, \sim D$, $\sim C \,\&\, \sim D$, $\sim C \vee \sim D$, $\sim C \vee D$, $(\sim C \vee D) \,\&\, (C \vee \sim D)$, $(C \vee D) \,\&\, (\sim C \vee \sim D)$, $(Ex) \sim B$, and $(Ax) \sim B$.

2) Replace subformulas of the form (Ax) B and (Ex) B by B if x is not free in B.

3) Replace subformulas of the form (Ax) . B (x) & C (x) and (Ex) . B (x) v C (x) respectively by (Ax) B (x) & (Ay) C (y) and (Ex) B (x) v (Ey) C (y) where y is

the first variable in alphabetic ordering not previously appearing.

4) Replace subformulas of the form $(Ex) . C \& D$, $(Ex) . D \& C$, $(Ax) . C \vee D$, and $(Ax) . D \vee C$ respectively by $(Ex) C \& D$, $D \& (Ex) C$, $(Ax) C \vee D$, and $D \vee (Ax) C$ where x is not free in D.

5) Replace subformulas of the form $(Ax) B(x)$ by $(Ay) B(y)$ if some quantifier binding x appears to the left of $(Ax) B(x)$, where y is the first variable in alphabetic order not previously appearing.

(Here and below the x's displayed in $B(x)$ are all the free occurrences of x.)

6) Replace subformulas of the form $(Ex) B(x)$ which is not in the scope of a universal quantifier by $B(a)$, where a is the first individual constant in alphabetic order not previously appearing.

7) Replace subformulas of the form $(Ex) B(x)$ by $B(G^{(n)}(y_1, \ldots y_n))$ where $y_1, \ldots, y_n$ are all and only the variables bound by universal quantifiers in whose scope this particular occurrence of $(Ex) B(x)$ appears, where $G^{(n)}$ is the first n-ary function constant not previously appearing.

8) Replace subformulas of the form $(Ax) B$ by B.

Example. Let us compute the Skolemization of

(Ex) (Ay) . (Ax) P (y,x) ⊃ (Ez) Q (z, x, y) . The steps are

a) (Ex) (Ay) . (Ex) ~ P (y, x) v (Ez) Q (z, x, y) , by 1) twice;

b) (Ay) . (Ex) ~ P (y, x) v (Ez) Q (z, a, y) , by 6);

c) (Ay) .~P (y, G(y)) v Q(H (y) , a, y) , by 7) twice; and

d) ~ P (y, G(y)) v Q (H(y) , a, y) , by 8),

where d) is the required Skolemization. Here and below let us assume that the symbols represented comply with the requirements of alphabetic ordering. Note that a) is equivalent to the original formula using the usual meaning of ⊃ , v , ~ , (Ex) and (Ax) .

In b) we have replaced the initially existentially quantified variable by a previously unused constant-- a device in common use in informal mathematics. In c) we have replaced x in ~ P (y, x) by G (y) where G (y) is to be the name of the object x which is said to exist for each choice of y . Since G did not appear previously, G (y) gives us no additional information beyond the existence the object for each y . Similar statements hold concerning H (y) . In d) we simply delete the universal quantifier. By this informal reasoning we see that a contradiction can be obtained from the original wff if and only if it can be obtained from its Skolemization d) . Examination of the definition of Skolemization leads us to make this assertion for general closed (no free variables) wffs.

In modifying the definition of a theorem for S_1 to serve for S_3 we can modify clause 1) so that $B_1^{(1)}$ is the Skolemization of $\sim(Ax_1)....(Ax_n)\,B$ where $x_1, ..., x_n$ are all the free variables of B. Clauses 2) and 4) can remain the same. Of course the reduction rules involving $\supset$ and $\equiv$ are not needed. In clause 3) we use atomic wffs in the role of the propositional variable p. Since the free variables occurring in the subformulas of the Skolemized wff were previously bound by the deleted universal quantifiers, we must allow ourselves all substitution instances of these quantifier-free wffs in our attempt to derive F. However, it is much too inefficient to blindly generate instances of these wffs. Suppose B and C are atomic formulas. Note that an instance of $\sim B \vee C$, call it $\sim B' \vee C'$, will be of use only if there is some previously generated wff D (or $\sim D$) and an instance D' of D (or $\sim D'$ of $\sim D$) such that D' is either B' or C'. So that rather than generate all possible $\sim B' \vee C'$'s and D''s it is much more efficient to consider $\sim B \vee C$ and D (or $\sim D$) to see if D and B (or D and C) have a common substitution instance (called a <u>matching formula</u> below). Further, it is important that this common instance be as general as possible. These ideas lead us to the notion of <u>matching</u>, which was developed for SAM II .

<u>Definition.</u> Two wffs of S_3 are said to <u>match</u> if they have some substitution instance (called a <u>matching formula</u>) in common. A matching

formula from which every other matching formula can be obtained by substitution is called a general matching formula.

Examples:

1) Q (a, x) and Q (x, H(x)) match and have Q (a, H(a)) as a general matching formula.

2) P (G (a, x)) and P (G(b, y)) do not match since the constants a and b do not match.

3) Q (x, x) and Q (y, H(y)) do not match since y and H(y) cannot be made identical by replacing y by the same wft in both y and H(y) .

4) Q (x, y) and Q (y, x) match and have Q (x, x) as a matching formula. Q (x, x) is not a general matching formula since, for example, Q (x, y) is a matching formula which is not an instance of Q (x, x) . Note that in this case Q (x,y) and Q (y, x) are both general matching formulas.

Definition. The following algorithm which is to be applied to two atomic wffs B and C of S_3 is called matching:

Step 1: Consider B and C as being stored at lines (1) and (2) respectively. Reletter the variables of line (2) so that it has no variables in common with line (1).

Step 2: Let us denote the n-th symbol -- ignoring parentheses and commas -- of line (1) by $(1)_n$. Similarly we define $(2)_n$.

Case a): If lines (1) and (2) are identical, the algorithm outputs (1) and stops.

Case b): Suppose n is the smallest integer such that $(1)_n$ is different from $(2)_n$. Since wffs are involved and case a) does not hold, neither $(1)_n$ or $(2)_n$ can be vacuous. We consider four subcases:

i) Suppose $(2)_n$ is a variable, say x, while $(1)_n$ is a function or individual constant. Then call D the unique subformula of (1) starting at $(1)_n$. If D contains x, output DOES NOT MATCH and stop. If D does not contain x, substitute D for x everywhere in (1) and (2). Go back and repeat step 2.

ii) Proceed as in i) if the roles of (1) and (2) are interchanged.

iii) If $(1)_n$ and $(2)_n$ are different variables, replace $(2)_n$ everywhere in (1) and (2) by $(1)_n$.

iv) If $(1)_n$ and $(2)_n$ are different constants, output DOES NOT MATCH and stop.

Examples. Let us apply matching to P(G(G(x, G(y,x)), z)) and P(G(G(x,y), G(x,y))).

(1) PGG x G y x z

(2) PGG uvG uu

(1) PGG x Gy x z

(2) PGG x v G x x

(1) PGG x G y x z

(2) PGG x G y x G x x

(1) PGG x Gy x G x x

(2) PGG x G y x G x x

Then P(G(G(X, G(y,x)) , G(x,x))) is the output of the algorithm and is in fact a general matching formula for the two wffs.

Let us apply matching to Q (x,x) and Q(y, H(y)) .

(1) Qxx

(2) Qy Hy

(1) Qxx

(2) Qx Hx

(1) DOES NOT MATCH

The variable x cannot be replaced by H(x) . This is a very simple example of a subtle way in which two wffs can fail to match. The more usual case of not matching arises from subcase iv) .

Theorem. If matching is applied to two atomic wffs B and C of S_3, then in a finite number of steps the algorithm outputs a general matching formula if a matching formula exists and outputs DOES NOT MATCH if no matching formula exists.

Proof: First we note that the process stops in a finite number of steps since each application of step 2 which does not stop the procedure eliminates a variable from (1) and (2). Let H(R) for a formula R be the hypothesis that there is a simultaneous substitution in (1) and (2) which yields two copies of R and further R is a general matching formula for (1) and (2).

Claim: H(R) holds before an application of step 2, case b) if and only if H(r) holds after step 2, case b).

Suppose subcase iii) is about to be applied and H(R) holds. Then the same formula must be substituted for the variables which occur at $(1)_n$ and $(2)_n$ in order to obtain two copies of R. Hence H(R) holds after one of these variables is replaced everywhere by the other. Conversely if H(R) holds after an application of subcase iii) then H(R) must have held before.

Suppose subcase i) or ii) is about to be applied and H(R) holds. If in the simultaneous substitution D is translated into D' then v must be replaced by D' by the substitution. Note that if D properly contains v then the simultaneous substitution cannot possibly yield identical formulas at (1) and (2) and hence H(R) would not

have held. if A does not contain v then H(R) will still hold after applying the subcase. The converse is similarly true.

If subcase iv) holds it is clear that H(R) for any R could not hold before or after applying iv).

Hence the claim is verified. If some formula R is outputed by step 2, case a) then using our claim R must be a general matching formula for B and C. Let H(R)' be the hypothesis H(R) with the word "general" deleted. By examining the proof of the claim we see that the modified claim -- H(R)' holds before step 2, case b) if and only if it holds after -- is similarly true. Now suppose DOES NOT MATCH is outputed. Then for no R does H(R)' hold for (1) and (2). By our modified claim H(R)' holds for no R for (1) and (2) immediately after step 1. But in this situation (1) and (2) have no variables in common so that we can conclude that B and C do not match. Q.E.D.

Theorem. If in the matching algorithm atomic wff B is written in (1) and atomic wff C is written in (2), then the algorithm outputs B if and only if B is an instance of C.

Proof: This is easily checked from the definition of matching.

Definition: We use matching to define a rule we call matching reduction: Let B and C be atomic formulas. If B is an instance of C then replace C (resp. ~ C) and D(...B...) by C (resp. ~ C) and D(...T...) (resp. D(...F...)). If B matches C with the general matching

formula B' but is not an instance of C , then add to C (resp. $\sim C$) and D(...B...), the formula D'(...T...), where D'(...B'...) is obtained from D(...B...) by making the same simultaneous substitution required to get B' from B (resp., add D'(...F...)) .

We are finally ready to define the class of theorems of S_3.

Definition. A finite sequence of finite sets of quantifier-free wffs of S_3 is said to be a proof of the theorem B if

1) first set contains only the Skolemization of $(Ex_1)....(Ex_n) \sim B$ where $x_1, ..., x_n$ are all and only the free variables of B ,

2) the final set contains the wff F ,

3) every set but the first is obtained from its immediate predecessor by a single application of matching reduction or one of the reduction rules 1), 4), 6) and 7) of S_1 .

Examples:

$\vdash_3$ (Ax) P (y,x) $\supset$ (Ex) P (x,z)

Proof: $\{ P(a,x) \;\&\; \sim P(x,b) \}$, $\{ P(a,x) , \sim P(x,b) \}$,
$\{ P(a,x) , \sim P(x,b) , \sim T \}$, $\{ P(a,x) , \sim P(x,b) , F \}$

$\vdash_3$ (Ay)(P(x) $\supset$ Q(y)) $\supset$. P(x) $\supset$ (Ay) Q(y)

Proof: $\{ (P(a) \supset Q(y)) \;\&\; (P(a) \;\&\; \sim Q(b)) \}$, ... ,
$\{ P(a) \supset Q(y) , P(a) , \sim Q(b) \}$, $\{ T \supset Q(y) , P(a) , \sim Q(b) \}$,
$\{ Q(y) , P(a) , \sim Q(b) \}$, $\{ F , P(a) , \sim Q(b) \}$.

$\vdash_3$ (Ax) P (x) $\supset$ P (y)

Proof: $\{ P(x) \ \& \sim P(a) \}$, $\{ P(x) , \sim P(a) \}$,

$\{ F , \sim P(a) \}$

Examples from Group Theory. Let GROUP be an abbreviation for the formula (commas are dropped for compactness)

(Ax) (Ay)(Az)(Au)(Av)(Aw) . P(exx) & P(I(x)xe) &

(P(xyu) $\supset$. P(uzw) $\supset$. P(yzv) $\supset$ P(xvw)) &

(P(yzv) $\supset$. P(xvw) $\supset$. P(xyu) $\supset$ P(uzw)) .

Then $\vdash_3$ GROUP $\supset$ P(xI(x)e)

Proof: Skolemization and several reductions gives us a set containing the formulas

(B_1) P(exx) ,

(B_2) P(I(x)xe) ,

(B_3) $\sim$ P(xyu) v. $\sim$ P(uzw) v. $\sim$P(yzv) v. P(xuw) ,

(B_4) $\sim$ P(yzv) v. $\sim$P(xvw) v. $\sim$ P(xyu) v. P(uzw) , and

(B_5) $\sim$ P(aI(a)e) .

Now by applying matching reduction and S_1 reductions we obtain the following additional formulas:

(B_6) $\sim$ P(yI(a)v) v. $\sim$ P(xve) v $\sim$ P(xya) , by applying (B_5) to (B_4) (written "B_5 into B_4") ,

(B_7) $\sim$ P(x I(a)e) v. $\sim$ P(xea) , by "B_1 into B_6 , "

(B_8) $\sim$ P(I(I(a))ea) , by "B_2 into B_7 , "

(B_9) $\sim$ P(I(I(a))yu) v. [illegible] $\sim$ P(yze) , by "B_8 into B_3 ,"

(B_{10}) $\sim P(eza) \vee \sim P(I(a)ze)$, by "B_2 into B_9 , "

(B_{11}) $\sim P(I(a)ae)$, by "B_1 into B_{10} , "

Now (B_{11}) is reduced to $\sim T$ and then F .

$\vdash_3$ GROUP $\supset$ P (xea)

Proof: Skolemization and several reductions gives as

(B_1) , (B_2) , (B_3) , (B_4) , and

(C_1) $\sim P(aea)$

Now we obtain

(C_2) $\sim P(ayu) \vee \sim P(uza) \vee \sim P(yze)$, by "C_1 into B_3 , "

(C_3) $\sim P(aye) \vee \sim P(yae)$, by "B_1 into C_2 , "

(C_4) $\sim P(aI(a)e)$, by "B_2 into C_3 . "

Now we can derive F from (B_1) , (B_2) , (B_3) , (B_4) , and (C_4) as was demonstrated in the previous proof.

The fact that one of the premisses of a matching reduction must be an atomic formula limits the power of S_3 though it greatly aids computational efficiency. For example the set of wffs

$\{ P_1(x) \ \& \ P_2(x) \vee Q_1(x) \ \& \ Q_2(x)$, $\sim P_1(y) \vee \sim P_2(y)$,

$\sim Q_1(z) \vee \sim Q_2(z) \}$

is contradictory but we can not reduce this set to include F by our rules of S_3 . Also S_3 --as did S_1 -- lacks sufficient devices to handle the propositional connectiv es. For example, no wffs of the forms

$B \equiv C$, $B \ \& \ C$, $\sim B \supset C$, or $\sim . B \vee C$ are theorems of S_3 .

Hence with the relationship between S_1 and S_2 in mind we extend S_3 to S_4 .

4. The Formal System S_4

The system S_4 is obtained from S_3 by modifying the definition of the proof of a theorem. S_4 is a complete system for the first order predicate-function calculus.

Definition. Let B* be $\sim(Ay_1) \ldots (Ay_n)\,((\sim(Ax_1) \ldots (A_n)B)^O)$, where $x_1, \ldots, x_h$ are all the free variables of B and $y_1, \ldots, y_m$ are all the free variables of $(\sim(Ax_1) \ldots (Ax_n)B)^O$.

Let J_b be the set of all variable-free terms which can be formed from the constants in B* (if B* contains no individual constants, let J_b be all the terms in the constants of B* and a -- the first individual constant in alphabetic order).

Let L_b be the set of all variable-free atomic wffs obtained by instantiating the y_i's in the atomic wffs of B* by elements of J_b.

Example. Let B be $\sim(Ex)(Ay) \,.\, (Ax)P(y,x) \supset (Ez)\,Q(z,x,y)$. Then B^O is $\sim P(y,G(y)) \vee Q(H(y), a, y)$ (see the first example of §3). Hence B* is $\sim(Ay) \,.\, \sim P(y,G(y)) \vee Q(H(y), a, y)$.

Now $J_b = \{ a, G(a), H(a), GG(a), GH(a), \ldots \}$

and $L_b = \{ P(M,G(M)) : M \in J_b \} \cup \{ Q(H(M), a, M) : M \in J_b \}$.

Theorems of S_4 : A finite sequence $B_1, \ldots, B_n$ of wffs of S_b is said to be a proof of B in S_4 if B_n is B* and if for each i $(i = 1,2,\ldots,n)$ B_i is a theorem of S_3 or there are j and k $(1 \leq j < i$ and $1 \leq k < i)$ such that for some C in L_b, B_j

is $C \supset B_i$ and B_k is $\sim C \supset B_i$.

Example.

$$\vdash_4 \quad (Ax)(P_1(X) \ \& \ P_2(x) \ \text{v.} \ Q_1(x) \ \& \ Q_2(x)) \supset (Ex).\ P_1(x) \ \& \ P_2(x) \ \text{v.} \ Q_1(x) \ \& \ Q_2(x) \qquad (B)$$

Proof: We find B* to be

$$\sim (Ax)(Ay)(Az) .\ (P_1(x) \ \& \ P_2(x) \ \text{v.} \ Q_1(x) \ \& \ Q_2(x)) \ \&. \ (\sim P_1(y) \ \text{v} \ \sim P_2(y)) \ \&. \ \sim Q_1(z) \ \text{v} \ \sim Q_2(z) \qquad (B^*)$$

Also J_b is $\{a\}$

and L_b is $\{P_1(a),\ P_2(a),\ Q_1(a),\ Q_2(a)\}$

It is easily checked that

$\vdash_3 \quad P_1(a) \supset B^*$ and $\vdash_3 \sim P_1(a) \supset B^*$.

Hence $\{P_1(a) \supset B^*,\ \sim P_1(a) \supset B^*,\ B^*\}$ is a proof of B in S_4 .

In an appendix to this section we shall outline a proof of the Godel - Herbrand-Gentzen theorem for S_4 (i.e., a wff of S_4 is a theorem if and only if it is valid in every (non-empty) model). Unlike the usual formulations of the first order predicate calculus, the completeness theorem and the accompanying proof-procedure follows directly from the definition of proof in S_4 . This is not to suggest that this proof-procedure can be mechanized to do any but the most trivial mathematics. This author firmly believes that this procedure and all others developed to date are not even remotely close to being useful as completely auto-

mated procedures. However our program of Semi-Automated Mathematics is much less ambitious (and hence more realistic). And here S_4 is quite useful; for while the mathematician-operator directs the over-all strategy of the proof, the machine can easily do some of the numerous, trivial, and bothersome steps since the proof-means of S_4 coincide so closely with its related proof-procedure.

Appendix to 4 . The following steps sketch a proof of the fact that $\vdash_4 B$ iff B is valid.

1) $\vdash_3 B$ implies B is valid since

   a) the skolemization of the negation of the universal closure of B is not satisfiable iff B is valid

   b) a set of wffs of S_3 is "satisfiable" iff it is after an application of one of the reduction rules of S_3 , and

   c) no set containing F is "satisfiable."

2) $\vdash_4 B$ implies B is valid, by 1).

3) B not provable in S_4 implies that for every finite subset $\{C_1, \ldots, C_n\}$ of J_b there is a non-theorem of S_3 of the form

$\tilde{C}_1 \supset \tilde{C}_2 \supset \ldots \supset \tilde{C}_n \supset B^*$, where $\tilde{C}_i$ is C_i or $\sim C_i$.

4) Let $C_1, C_2, C_3, \ldots$ be an enumeration of J_b . By 3) there is a sequence of non-theorems of S_3 of the form

$C_1^{(1)} \supset B^*$, $C_1^{(2)} \supset C_2^{(2)} \supset B^*$, $C_1^{(3)} \supset C_2^{(3)} \supset C_3^{(3)} \supset B^*$,

$\ldots$ where $C_j^{(i)}$ is C_j or $\sim C_j$. Let D_1 be C_1 if infinitely many of the $C_1^{(i)}$'s are C_1 . Let D_1 be $\sim C_1$ otherwise. Of the infinitely many i's such that $C_1^{(i)}$ is D_1, if infinitely many have $C_2^{(i)}$ being C_2 then let D_2 be C_2 ; otherwise let D_2 be $\sim C_2$. And so on. In this way we have picked $D_1, D_2, D_3, \ldots$ so that for each finite n , infinite many of the non-theorems in the sequence

above start with $D_1 \supset D_2 \supset \cdots \cdot D_n \supset \cdots$.

Hence for each n, $D_1 \supset \cdots \cdot D_n \supset B^*$ is not provable in S_3.

5) From 4) we can define a model whose elements are J_b in which B* does not hold. The values of the functions of B* are given by the rule: the value of $G^{(n)}$ when applied to the elements $M_1, \ldots, M_n$ of J_b is to be the element $G^{(n)}(M_1, \ldots, M_n)$ of J_b. The values of the predicates of B* are given by: the value of $P^{(n)}$ when applied to the elements $M_1, \ldots, M_n$ of J_b is to be T if some D_i is $P^{(n)}(M_1, \ldots, M_n)$ and F otherwise.

6) B does not hold in the model defined in 5). This follows easily from the definition of B* and 5).

5. The Formal System S_5

The system S_5 is to be an ω-order predicate-function calculus patterned after S_4. Enough type mechanism is included in S_5 to differentiate between predicates and general functions while allowing certain function-like properties of predicates to be obtained as instances of more general theorems about functions. Again we postpone to a later report, the discussion of equality.

Primitive symbols of S_5 :

1) Improper symbols $\langle \quad \rangle \quad (\quad) \quad [\quad] \quad ,$

2) Logical symbols $\supset \quad \sim \quad \vee \quad \& \quad \equiv \quad \Pi \quad \Sigma \quad \lambda \quad T \quad F$

3) Type constants $\quad U \quad V$

4) An infinite list of variables of each type.

5) An infinite list of constants of each type.

Type formalism:

1) V is a type (– the truth-value type).

2) U is a type (– the type of the universe).

3) If τ_1, τ_2, ..., τ_n and τ are types, then $\langle \tau_1, \ldots, \tau_n, \tau \rangle$ is a type (– the type of all n-ary functions whose n arguments have the types $\tau_1, \ldots, \tau_n$ respectively and whose value is type τ.

Meta-language for S_5 :

1) τ , σ , δ , ρ , τ_1, σ_1 , ... to range over types.

2) Lower case roman letters with integer subscripts and type superscripts to range over variables of that type.

3) G^{τ} , H^{τ} , M^{τ} , N^{τ} , G_1^{τ} , H_1^{τ} , ... to range over constants of type τ .

4) B^{τ} , C^{τ} , D^{τ} , B_1^{τ} , C_1^{τ} , ... to range over terms of type τ .

Type containment : Type τ is said to be contained in type σ , written $\tau \subset \sigma$, as defined by the following inductive definition:

1) $V \subset U$

2) $\tau \subset \tau$

3) $\tau_1 \subset \tau_2$ and $\tau_2 \subset \tau_3$ implies $\tau_1 \subset \tau_3$.

4) If $\sigma_i \subset \tau_i$ $(i = 1, \ldots, n)$ and $\tau \subset \sigma$ then $\langle \tau_1, \ldots, \tau_n, \tau \rangle \subset \langle \sigma_1, \ldots, \sigma_n, \sigma \rangle$.

We define $\max(\tau, \sigma)$ and $\min(\tau, \sigma)$ by

1) $\max(V, U) = \max(U, V) = U$,

2) $\min(V, U) = \min(U, V) = V$,

3) if $\min(\tau_i, \sigma_i)$ and $\max(\tau, \sigma)$ have been defined then

$$\max(\langle \tau_1, \ldots, \tau_n, \tau \rangle, \langle \sigma_1, \ldots, \sigma_n, \sigma \rangle) = \langle \min(\tau_1, \sigma_1), \ldots, \min(\tau_n, \sigma_n), \max(\tau, \sigma) \rangle ,$$

4) if $\max(\tau_i, \sigma_i)$ and $\min(\tau, \sigma)$ have been defined then

$$\min(\langle \tau_1, \ldots, \tau_n, \tau \rangle, \langle \sigma_1, \ldots, \sigma_n, \sigma \rangle) = \langle \max(\tau_1, \sigma_1), \ldots, \max(\tau_n, \sigma_n), \min(\tau, \sigma) \rangle$$

5) $\min(\tau, \sigma)$ and $\max(\tau, \sigma)$ are left undefined unless their being defined follow from 1) - 4).

Notice that $\min(\tau, \sigma)$ is the most general common type of τ and σ, if τ and σ have a common type.

Well-formed expressions of S_5

1) T and F are well-formed terms (wfts) of type V.

2) $\supset$, &, v, $\equiv$ are wfts of type $\langle V, V, V \rangle$.

3) $\sim$ is a wft of type $\langle V, V \rangle$.

4) A constant or variable of type τ is a wft of type τ.

5) If $\sigma_i \subset \tau_i$ $(i = 1, \ldots, n)$ then $[B \langle \tau_1, \ldots, \tau_n, \tau \rangle] (B_1^{\sigma_1}, \ldots, B_n^{\sigma_n})$ is a wft of type τ.

6) $(\lambda x_1^{\tau_1}, \ldots, x_n^{\tau_n}) B^{\tau}$ is a wft of type $\langle \tau_1, \ldots, \tau_n, \tau \rangle$. The occurrences of x in this wft are said to be bound. The free occurrences of x^{τ_i} in B^{τ} are said to be bound by the initial occurrence of λ in $(\lambda x_1^{\tau_1}, \ldots, x_n^{\tau_n}) B^{\tau}$.

7) $(A\, x^{\tau})\, B^{V}$ and $(E\, x^{\tau})\, B^{V}$ are wfts of type V.

The definitions of bound and free are analogous with S_3.

8) A wft of type V is called a well-formed formula (wff).

Substitution. We use the following substitution notation in the meta-language of S_5: If $\sigma \subset \tau$, then

$$S^{x^{\tau}}_{B^{\sigma}} \; C^{\zeta} \mid$$

stands for the result of replacing all the free occurrences of x^{τ} in C^{ζ} by B^{σ} and relettering (in some canonical fashion) those bound variables of C^{ζ} which would otherwise capture some free variables of B^{σ}.

Lambda-conversion. In forming the Skolemizations and as a reduction rule for S_5 we use the following rule of λ-conversion: A wft of the form $[(\lambda x_1, \ldots, x_n)\, B]\, (C_1, \ldots, C_n)$ can be replaced by $S^{x_1 \cdots x_n}_{C_1 \cdots C_n} B \mid$. A wft is said to be λ-converted if λ-conversion is not applicable to any sub-wfts of the wft.

Special-conversion. A wft of the form $(\lambda x_i, \ldots, x_n)([B](x_1, \ldots, x_m))$ where the x_i's do not appear free in B, can be replaced by B, and conversely.

Matching. Any wft C obtained from a wft B by finitely many applications of substitution, relettering bound variables, λ-conversions, and

special-conversions is called an instance of B . Two wfts B_1 and B_2 are said to match if they have a common instance (called a matching term for B_1 and B_2) . A matching term for B_1 and B_2 , from which every other matching term can be obtained, is called a general matching term.

Example: Unlike S_4 , two wfts of S_5 can match without having a general matching term. For example G (f (x) , f , x) and G (H (M), f , x) have as matching terms G(H (M), H , M) , G(H(M) , (λ z) H(M) , x) , and G(H(M) , (λ z) z , H(M)). However G (f (x) , f, x) and G(H(M), f, x) have no general matching term as can easily be seen. Still we notice that G(H(M) , (λ z) H(M) , x) is a more general term than G(H(M) , (λ z) H(M) , N) though both are matching terms for the original two formulas. This leads us to the following:

Definition. A general matching set for wfts B_1 and B_2 is a set of matching terms for B_1 and B_2 such that every other λ -converted matching term is an instance of some member of the set and no member of the set is an instance of another member. Two terms for which neither is an instance of the other are called independent.

In order to describe an algorithm for matching in S_5 we first define some concepts.

Definition. We define the character of a sub-wft of a wft B as follows:

1) A constant has constant character.

2) A free variable of B has variable character.

3) A bound variable of B has constant character.

4) (Ax) B and (Ex) B have constant character.

5) $[B]\ (B_1, \ldots, B_n)$ and $(\lambda x_1 \ldots, x_n)P$ have the same character as B.

Definition. Two λ-converted wfts are said to rough-match under the following inductive definition:

1) Two wfts B^{τ} and C^{σ} of variable character rough-match if their types contain a common type; i.e. if $\min(\tau, \sigma)$ is defined.

2) A wft B^{τ} of constant character rough-matches a wft C^{σ} of variable character if $\tau \subset \sigma$.

3) Two identical constants rough-match.

4) Two bound variables rough-match if they have the same type.

5) Two wfts of constant character $[B_o]\ (B_1, \ldots, B_n)$ and $[C_o]\ (C_1, \ldots, C_n)$ rough-match if B_i and C_i rough-match $(i = o, 1, \ldots, n)$.

6) Two wfts $(Ax^{\tau})\ B$ and $(Ay^{\sigma})\ C$ of constant character rough-match if $\tau = \sigma$ and if B and C rough-match. Similarly for (Ex) and $(\lambda x_1 \ldots, x_n)$. (Special-conversion may have

to first be applied to a wft of constant character which corresponds to a wft of constant character whose first proper symbol is λ .)

Remark: If two wfts do not rough-match then they do not match. Two wfts without free variables or quantifiers rough-match if they match.

We shall use a series of examples to describe the matching algorithm for S_5 . Once the matching algorithm has been defined, we can define the concept of proof in S_5 analogously to that in S_4 (going first to an intermediate system S_5' which corresponds to S_3) . Of course the theorems of S_5 will be valid in every non-empty model, but the converse of this result (The Henkin Completeness Theorem) has no known proof which corresponds to the Godel-Herbrand-Gentzen method. Hence we have no appendix for this section corresponding to the appendix of Section 4.

Example: Let x, y, z, x_1, M, N be objects of type τ ; let f, f_1, G be objects of type $\langle \tau, \langle \tau, \sigma \rangle \rangle$ where $\sigma \notin \tau$: and let g and h be objects of type $\langle \tau, V \rangle$. Remember that the lower case letters are variables and the upper case letters are constants. Let us match

(1) $H([f_1(x)](z), z, f_1, (Ah)\ h(x))$, and

(2) $H([G(N)](y), M, f, (Ag)g(y))$.

If necessary we would reletter the free and bound variables of (1) and (2) so that no free variable occurs in both (1) and (2) and such that no variable appears both bound and free in (1) and (2). Now we proceed from left-to-right making the more routine matches. For example we by-pass the corresponding sub-wfts $[f_1(x)](z)$ and $[G(N)](y)$ since f_1 is a free function variable with arguments and hence it is not necessary that x match N or z match y. We do the routine matches first so that the matching of free function variables with arguments will be as simple as possible. In this spirit we match z with M and f_1 with f resulting in

(1.1) $H([f(x)](M), M, f, (Ah)\ h(x))$.

and (2.1) = (2). In general when we are matching variables f^{τ} and f_1^{σ} which appear without arguments, we substitute f^{τ} for f_1^{σ} if $\tau \subset \sigma$, f_1^{σ} for f^{τ} if $\sigma \subset \tau$, $f_2^{\min(\sigma,\tau)}$ for f_1^{σ} and f^{τ} (where $f_2^{\min(\sigma,\tau)}$ has not previously appeared) if $\min(\sigma,\tau)$ is defined, and finally we do not get a match if σ and τ have incompatible types -- i.e. $\min(\sigma,\tau)$ not defined. Now we consider the corresponding sub-wfts (Ah) h(x) and (Ag) g (y). If the corresponding bound variables g and h were not

the same type we would not have a match. We reletter the bound variables so that the corresponding bound variables are the same. This results in (2.2) = (2.1) and

(1.2) H([f(x)] (M) , M , f , (Ag) g (x)) .

Interior bound variables are considered as constants so that x and y in (Ag) g (x) and (Ag)g (y) must match resulting in (1.3) = (1.2) and

(2.3) H([G(N)] (x) , M , f , (Ag) g (x)) .

Now we have to match [f(x)] (M) and [G(N)] (x) . Notice that this should be considerably easier than attempting to match these sub-wfts initially -- for we have identified two variables and replaced a third variable by a constant. We wish to consider every substitution for f of the form (λu) (λ v) B (u, v) where [[(λu) (λ v) B (u, v)] (x)] (M) , or equivalently after two applications of lambda-conversion, B(x, M) , matches [G(N)] (x) . However we consider all substitutions of this form so that B(x, M) rough-matches [G(N)] (x) , using the algorithm itself to eliminate later any substitutions which lead to non-matching. The B (u , v) 's , which we call candidates, are defined inductively by

1) [G(N)] (x) is a candidate ; and

2) if B' is a candidate and some sub-wft B" of B' rough-matches x (respectively M) , then the result of replacing

B" in B' by u (respectively v) is also a candidate. Hence we have the six candidates $[G(N)](x)$, $[G(N)](u)$, $[G(N)](v)$, $[G(u)](x)$, $[G(u)](u)$, and $[G(u)](v)$. After substitution and lambda-conversion we obtain the six matching problems

(1-1) $H\ ([G(N)](x),\ M,\ (\lambda w)(\lambda v)[G(N)](x),\ (Ag)\ g\ (x))$

(2-1) = (1-1)

(1-2) $H\ ([G(N)](x),\ M,\ (\lambda u)(\lambda u)[G(N)](u),\ (Ag)\ g\ (x))$

(2-2) = (1-2)

(1-3) $H\ ([G(N)](M),\ M,\ (\lambda u)(\lambda v)[G(N)](v),\ (Ag)\ g\ (x))$

(2-3) $H\ ([G(N)](x),\ M,\ (\lambda u)(\lambda v)[G(N)](v),\ (Ag)\ g\ (x))$

(1-4) $H\ ([G(x)](x),\ M,\ (\lambda u)(\lambda v)[G(u)](x),\ (Ag)\ g\ (x))$

(2-4) $H\ ([G(N)](x),\ M,\ (\lambda u)(\lambda v)[G(u)](x),\ (Ag)\ g\ (x))$

(1-5) $H\ ([G(x)](x),\ M,\ (\lambda u)(\lambda v)[G(u)](u),\ (Ag)\ g\ (x))$

(2-5) $H\ ([G(N)](x),\ M,\ (\lambda u)(\lambda v)[G(u)](u),\ (Ag)\ g\ (x))$

(1-6) $H\ ([G(x)](M),\ M,\ (\lambda u)(\lambda v)[G(u)](v),\ (Ag)\ g\ (x))$

(2-6) $H\ ([G(N)](x),\ M,\ (\lambda u)(\lambda v)[G(u)](v),\ (Ag)\ g\ (x))$

We consider these problems to be in a push-down list. Let us treat the top problem, (1-6) and (2-6). We must match x and N. This results in replacing x everywhere in (1-6) and (2-6) by N.

But then we must match M with N , which is impossible. Hence we delete (1-6) and (2-6) from our list. Now we again treat the top problem, (1-5) and (2-5) . Here x must be replaced everywhere by N yielding

(1-5.1) $H([G(N)] (N) , M , (\lambda u) (\lambda v) [G(u)] (u) , (Ag) g (N))$

and (2-5.1) = (1-5.1) . Now we let our general matching set contain the term (1-5.1). Let us write this as GMS = $\{$(1-5.1)$\}$. Now we delete (1-5.1) and (2-5.1) from our list of problems. Next we obtain

(1-4.1) $H([G(N)] (N) , M , (\lambda u) (\lambda v) [G(u)] (N) , (Ag) g (N))$

and (2-4.1) = (1-4.1) . Since $(\lambda u) (\lambda v) [G(u)] (N)$ is independent of $(\lambda u) (\lambda v) [G(u)] (u)$, (1-4.1) is independent of (1-5.1) . Hence we add (1-4.1) to GMS getting GMS = $\{$(1-5.1) , (1-4.1)$\}$. Similarly we add (1-3.1), (1-2), (1-1) to GMS where

(1-3.1) $H([G(N)] (M) , M , (\lambda u) (\lambda v) [G(n)] (v) , (Ag) g (M))$.

Thus our final result is

$$GMS = \{(1-1) , (1-2) , (1-3.1) , (1-4.1) , (1-5.1)\}$$

Notice that if the third argument of H were deleted that GMS would contain only $H([G(n)] (x) , M , (Ag) g (x))$.

This result could be obtainted very quickly by noticing that

$H(w^{\sigma}, M, (Ag)\, g(x))$ and

$H([f_1(x)](M), M, (Ag)\, g(x))$

are instances of one another (using $(\lambda u)(\lambda v)\, w^{\sigma}$ for f_1).

This suggests the following rule: If every occurrence of a variable $f^{<\tau_1,\ldots,\tau_n,\tau>}$ throughout the two terms to be matched is with the same arguments and is never in the scope of a quantifier or λ, then f with its arguments can be replaced by w^{τ} where w^{τ} has not previously appeared. As an application of this rule, consider the following.

Example. We wish to generate GMS for the wfts

(1) $G(f(x), g(y, f), x)$

(2) $G(f_1(x_1), z, x_1)$

Then using the rule, $g(y, f)$ and $f_1(x_1)$ can be replaced by variables. Notice that we could not replace $f(x)$ by a variable at first, since f appears without arguments in $g(y, f)$. We then arrive at

(1.1) $G(w_3, w_1, x)$

(2.1) $G(w_2, z, x_1)$

These can now be matched in a straight-forward manner, the ultimate result depending on the types of the variables present.

If $u_1, \ldots, u_n$ do not occur free in D, then we can interchange $(\lambda u_1, \ldots, u_n)\ [D]\ (u_1, \ldots, u_n)$ with D using special-conversion. This raises a problem when $(\lambda u_1, \ldots, u_n)$ C must match D and the first proper symbol of D is not λ. For C may have the form $[D]\ (u_1, \ldots, u_n)$, or may be put in that form by making certain substitutions for free function variables with arguments. This situation is easily resolved by replacing D in these instances by $(\lambda u_1, \ldots, u_n)\ [D]\ (u_1, \ldots, u_n)$ and carrying on with the matching. For when we match C and $[D]\ (u_1, \ldots, u_n)$ we will find if C can be put in the appropriate form.

We have not encountered the situation in the examples considered thus far, where we are matching two sub-wfts whose first proper symbols are free function variables with arguments. We consider first an example where the first symbols are the same. In the second example we show how the general case can be reduced to the current case.

Example. Suppose after the beginning stages of matching we arrive at the situation:

(1) H (f(x, y, M), f, g, x, y, z)

(2) H (f(x, z, N), f, g, x, y, z)

Now we attempt to match f(x, y, M) and f(x, z, N).
We keep the corresponding arguments which are identical; we delete the corresponding arguments which don't rough-match; and then we

delete all 2^n possible combinations of the remaining n arguments (in this example $n = 1$), obtaining 2^n matching problems in place of the previous one. In this case we obtain

(1-1) $H(\tilde{f}_1(x, y), (\lambda u, v, w) f_1(u, v), x, y, z)$

(2-1) $H(\tilde{f}_1(x, z), (\lambda u, v, w) f_1(u, v), x, y, z)$

(1-2) $H(\tilde{f}_2(x), (\lambda u, v, w) f_2(u), x, y, z)$

and (2-2) = (1-2). The tildas ($\sim$) above the f_1 and f_2 indicate that this process has already been performed to this occurrence of the variable. For the purposes of matching the arguments inside the f 's, we consider the f's as constants. Hence from (1-1) and (2-1) we obtain

(1-1.1) $H(f_1(x, y), (\lambda u, v, w) f_1(u, v), x, y, y)$

and (2-1.1) = (1-1.1). Then

GMS = $\{$ (1-2), (1-1.1) $\}$.

Example. Consider the intermediate stage

(1) $G(f(M, H), f, g)$

(2) $G(g(F, M), f, g)$

Considering f as a constant, we get the following trial substitutions for g by the method of our first matching example:

$(\lambda u, v) f(M, H)$ and $(\lambda u, v) f(v, H)$.

These yield

(1-1) $G(f(M, H), f, (\lambda u, v) f(M, H))$

(2-1) $=$ (1-1)

(1-2) $G(f(M, H), f, (\lambda u, v) f(v, H)$

(2-2) $=$ (1-2)

Considering g as a constant, we get the following trial substitutions for f : $(\lambda u, v) g(F, M)$ and $(\lambda u, v) g(F, u)$. These yield

(1-3) $G(g(F, M), (\lambda u, v) g(F, M), g)$

(2-3) $=$ (1-3)

(1-4) $G(g(F, M), (\lambda u, v) g(F, u), g)$

(2-4) $=$ (1-4) .

Hence GMS is one element from each of these pairs, since they are independent. In general we would not be done this quickly, but we would be able now to apply the method of the previous example to complete the matching.

These examples suggest in an obvious way (we hope) an algorithm for generating a general matching set for two wfts. It is clear that the terms in GMS are matching terms for the two wfts. A detailed proof

that GMS is actually a general matching set, has not been carried out. There seems little point to that task until methods for handling substitutivity of equality have been added.

Remark. The following derived rule holds for S_5 : If we consider the types of the atomic symbols as being appended as superscripts, then the result of simultaneously replacing every occurrence of U in a theorem of S_5 by some fixed type expression, is again a theorem of S_5 . As an application of this rule we can obtain, for example, from the group theory concerning individuals, the group theory of other mathematical entities such as transformations, permutations, etc.

6. The Formal System S_6

The system S_6 is an n-order predicate calculus which is more powerful than S_5 in that a mechanism for defining sorted variables is included within the formation of S_6. This system suggests solutions to many of the problems which arise in considering the automation of a "many-sorted" system.

Primitive Symbols of S_6. The primitive symbols of S_6 are

$\supset$, $\sim$, $\vee$, $\&$, $\equiv$, $\forall$, $\exists$, λ , τ , $\bar{\subseteq}$,

U , V , T, F ,] , [, / , > , < ,) , (, and ,,

an infinite list of variable and constant forms.

Well-formed expressions of S_6.

1) Types and type containment are defined as in S_5.

2) U is a sort of type U .

3) V is a sort of type V .

4) If $\sigma_1, \ldots, \sigma_n, \sigma$ are sorts of type $\delta_1, \ldots, \delta_n, \delta$ respectively, then $\langle \sigma_1, \ldots, \sigma_n, \sigma \rangle$ is a sort of type $\langle \delta_1, \ldots, \delta_n, \delta \rangle$.

5) If x is a variable form and σ is a sort, then x/σ is a variable of sort σ.

6) If G is a constant form and σ is a sort, then G/σ is a constant of sort σ.

7) T and F are well-formed terms (wfts) of sort V .

8) The variables and constants of sort σ are wfts of sort σ.

9) If B is a wft of sort V and σ is a sort of type δ, then $(\tau\ x/\sigma)B$ is a sort of type δ. (In the intended interpretation, this sort, for fixed values of the other free variables of B besides x/σ, is to be the sort--or set-- of all elements of sort σ which satisfy $B(x/\sigma)$.)

10) If $B_1, \ldots, B_n, B$ are wfts of the respective sorts $\sigma_1, \ldots, \sigma_n, \langle \alpha_1, \ldots, \alpha_n, \alpha \rangle$ which have the respective types $\delta_1, \ldots, \delta_n, \langle \beta_1, \ldots, \beta_n, \beta \rangle$, and if $\delta_i \subset \beta_i$, then $[B](B_1, \ldots, B_n)$ is a wft of sort α.

11) If B is a wft of sort σ, then $(\lambda x_1/\sigma_1, \ldots, x_n/\sigma_n)B$ is a wft of sort $\langle \sigma_1, \ldots, \sigma_n, \sigma \rangle$. If B is a wft of sort V, then so are $(Ex/\sigma)B$ and $(Ax/\sigma)B$. (Free occurrences of x/σ in the sorts within B are considered to be bound by the initial occurrences of λ, A, E, and τ.)

12) If σ and δ are sorts, then $\sigma \tilde{\subseteq} \delta$ is a wft of sort V.

13) If B and C are wfts of sort V, then so are $\sim B$, $(B \supset C)$, $(B \vee C)$, $(B \& C)$, and $(B \equiv C)$.

We shall use a metalanguage for S_6 which is analogous to that of S_5.

We develop S_6 in a more standard fashion than the previous five systems in that we proceed from axioms by rules of inference. We

only sketch this development.

Axioms and Rules of S_6 . In addition to the usual rules and axioms needed to handle the logic, S_6 has two axiom schema for substitution:

A1) $\sigma \subseteq \delta \supset .(A x/\delta) B(x/\delta) \supset B(C)$, where C is a wft of sort σ ; and

A2) $(A y/(\tau x/\sigma) D(x/\sigma)) B(y/-) \supset . D(C) \supset B(C)$, where here and below $y/-$ stands for y/δ if δ is the last sort which appeared with y ; hence in this instance $y/-$ stands for $y/(\tau x/\sigma) D(x/\sigma)$.

Also S_6 has six axiom schema concerning sort containment:

A3) $V \subseteq U$;

A4) $\sigma \subseteq \sigma$;

A5) $\sigma_1 \subseteq \delta_1 \supset . \ldots . \supset . \sigma_n \subseteq \delta_n \supset . \delta \subseteq \sigma \supset \langle \delta_1, \ldots, \delta_n, \delta \rangle \subseteq \langle \sigma_1, \ldots, \sigma_n, \sigma \rangle$;

A6) $(Ex/\sigma) B \supset (\tau x/\sigma) B \subseteq \sigma$

A7) $\sigma_1 \subseteq \sigma_2 \supset . \sigma_2 \subseteq \sigma_3 \supset . \sigma_1 \subseteq \sigma_3$;

$(Ax/\sigma)(B \supset C) \supset . (Ex/\sigma) B \supset (\tau x/\sigma) B \subseteq (\tau x/\sigma) C$; and

A9) $(E y/\alpha) B(y/\alpha) \supset . B(x/(\tau y/\alpha) B(y/\alpha))$.

Analogous to the situation in S_5 , we will need to have the most general sort contained in both σ and δ -- abbreviated

by $glb(\sigma, \delta)$ --- in order to match x/σ and y/δ .

We define the abbreviations $glb(\sigma, \delta)$ and $lub(\sigma, \delta)$ simultaneously by

1) $glb(\sigma, U) = glb(U, \sigma) = \sigma$,

2) $lub(\sigma, U) = lub(U, \sigma) = U$,

3) if the abbreviations $lub(\sigma_i, \delta_i)$ and $glb(\sigma, \delta)$ have been defined, then define

$$glb(\langle \sigma_1, \ldots, \sigma_n, \sigma \rangle, \langle \delta_1, \ldots, \delta_n, \delta \rangle)$$
$$= \langle lub(\sigma_1, \delta_1), \ldots, lub(\sigma_n, \delta_n), glb(\sigma, \delta) \rangle ,$$

4) if the abbreviations $glb(\sigma_i, \delta_i)$ and $lub(\sigma, \delta)$ have been defined, then define

$$lub(\langle \sigma_1, \ldots, \sigma_n, \sigma \rangle, \langle \delta_1, \ldots, \delta_n, \delta \rangle)$$
$$= \langle glb(\sigma_1, \delta_1), \ldots, glb(\sigma_n, \delta_n), lub(\sigma, \delta) \rangle ,$$

5) if $glb(\sigma, \delta)$ has been defined, then let

$$glb((\tau x/\sigma) B (x/\sigma), (\tau y/\delta) C (y/\delta))$$
$$= (\tau z/glb(\sigma, \delta)) . B(z/-) \& C(z/-), \text{ and}$$

6) if $lub(\sigma, \delta)$ has been defined, then let

$$lub((\tau x/\sigma) B (x/\sigma), (\tau y/\delta) C (y/\delta))$$
$$= (\tau z/lub(\sigma, \delta)) . ((Ex/\sigma) . B(x/\sigma) \& x/\sigma = z/-)$$
$$\vee ((Ey/\delta) . C(y/\delta) \& y/\delta = z/-)$$

In order to demonstrate some of the advantages of S_6 for semi-automated mathematics, let us develop group theory in S_6 . To facilitate

this let us use s with integer subscripts and free variables displayed, to abbreviate certain sorts. We use $s_i(-)$ to stand for s_i with the arguments displayed in its last occurrence. Let s_1, s_2, and s_3 stand for $\langle U, V \rangle$, $\langle U, U, U \rangle$, and $\langle U, U \rangle$ respectively. Then let

GROUP $(g/s_1, h/s_2, e/U, i/s_3)$ stand for the wft

$$[g/s_1]\,(e/U)\ \&\ (Ax/U)\,(Ay/U)\,(Az/U).$$
$$\{[g/s_1]\,(x/U)\ \&\ [g/s_1]\,(y/U) \supset [g/s_1]([h/s_2]\,(x/U, y/U))$$
$$\&\ [g/s_1]\,([i/s_2]\,(x/u))\}$$
$$\&\ \{[h/s_1]\,(e/U, x/U) = x/U\}$$
$$\&\ \{[h/s_2]\,([i/s_3]\,(x/U), x/U) = e/U\}$$
$$\&\ \{[h/s_2]\,([h/s_2]\,(x/U, y/U), z/U)$$
$$= [h/s_2]\,(x/U, [h/s_2]\,(y/U, z/U))\}$$

Now we let $s_4(h/s_2, e/U, i/s_3)$ stand for $(\tau g/s_1)$ GROUP $(g/s_1, h/s_2, e/U, i/s_3)$. Then for example, $s_4(H, E, I)$ is the sort of all sets which are a group under the composition H, the identity E, and the inverse I. The sort of all elements of a group g belonging to $s_4(h/s_2, e/U, e/s_3)$ is written as $s_5(g/s_4(-))$ and is defined by $(\tau x/U)\,[g/-]\,(x/U)$.

In order to state theorems compactly we let $*B$ stand for $(Ex_1/\sigma_1)\,B_1\ \&\ \ldots\ \&\ (Ex_n/\sigma_n)\,B_n \supset B$ where $(\tau x_1/\sigma_1)\,B_1, \ldots, (\tau x_n/\sigma_n)\,B_n$ are all and only the distinct sort expressions

in B of the form $(\tau \ldots .)$. We are now ready to sketch the proof that the left identity of the group axioms is also a right identity. Notice that the group axioms do not have to be explicity stated.

$\vdash_6 * [h/s_2]\ (x/s_5\ (-)\ ,\ e/U) = x/-$.

Proof: The antecedent of this alleged theorem is

$(E\ y / U)\ g/s_4\ (-)\ (x/U)\ \&\ (Ef/s_1)$ GROUP $(f/s_1, h/s_2, e/U, i/s_3)$.

Now using the deduction theorem which holds in S_6 we have

1) ante. $\vdash$ GROUP $(g/-, h/s_2, e/U, i/s_3)$ by axiom A9.

2) ante. $\vdash [g/-]\ (x/s_5\ (-))$ by axiom A9.

Now by standard techniques, the proof can easily be completed using 1) and 2).

Now we are in a position to appreciate one of the advantages of S_6. Suppose we wish to replace $x/-$ in the theorem just proved by a constant $M/s_5\ (G/s_4\ (H, E, I))$. Before this substitution can be made, we have to first replace h, e, i, and g by H, E, I, and G respectively. When S_6 is (semi-) automated these four initial substitutions would be carried out automatically. Hence S_6 suggests a very straight-forward way of handling a very sticky problem of automating many-sorted ω-order calculi.

Using sorted variables, it is quite convenient to define new sorts. E.g. we let $s_6\ (g/s_4\ (h/s_2, e/U, i/s_3))$ stand

for $(\tau\ f/s_4(-))(Ax/s_5(f/-))\ [g/s_4(-)](x/-)$. Hence $s_6(g/-)$ is the sort of all subgroups of g. As a final example we see that COSET $(x/s_5(g/-),\ f/s_6(-))$ can be abbrev iated by $(\lambda\ y/s_5(-))\ [f/s_6(-)]\ ([h/s_2]\ ([i/s_3](x/-),\ y/-))$.

Semi-automation of S_6: We close this report with some comments and ideas bearing on the mechanization of S_6.

1) By using Skolemization, matching, and various reduction rules, a contradiction will be sought from the denial of a wft of S_6 and previously assumed steps, theorems, and axioms in a manner analogous to S_5.

2) In matching we might make a concession to practicality by not using $glb(\sigma, \delta)$ in matching x/δ and y/σ. Rather we might attempt to find instances δ' and σ' of δ and σ respectively such that $\delta' \equiv \sigma'$ or $\sigma' \equiv \delta'$.

3) Each distinct sort will be stored in a location and only the address of this location will be attached to the various symbols. The variables in σ which are free in σ but bound in the occurrence of σ in some wft, are stored free in σ. The variables in σ which are bound in σ are canonicalized to reduce the number of sorts stored. When substitutions are made or bound variables are re-lettered in wfts, new sorts may be generated.

4) A library of sort containments might be kept to increase the power of substitution and matching. Sort containments arising from implications being proved (see axiom A8) might be catalogued. Alternatively it might be simpler to search the theorem library for suitable implications.

BIBLIOGRAPHY

[1] F. B. Fitch, Symbolic Logic, New York, 1952.

[2] S. Jaskowski, "On the rules of supposition in formal logic," Studia Logica, no. 1, Warsaw, 1934.

[3] J. H. Bennett, W. B. Easton, J. R. Guard, T. H. Mott, Introduction to Semi-automated Mathematics: Section One: A Programming Language for Natural Deduction, AFCRL 63-188 (Contract No. AF 19 (628) - 468), 1963.

[4] J. H. Bennett, W. B. Easton, J. R. Guard, T. H. Mott, Semi-automated Mathematics: SAM III, report in preparation for Air Force Cambridge Research Laboratories (Contract No. AF 19 ( ) -).

[5] J. H. Bennett, W. B. Easton, J. R. Guard, T. H. Mott, Introduction to Semi-automated Mathematics: Section Two: Axiomatic Theorem Proving, AFCRL 63-188 (Contract No. AF 19 (628) - 468), 1963.

[6] S. C. Kleene, Introduction to Metamathematics, New York, 1952.